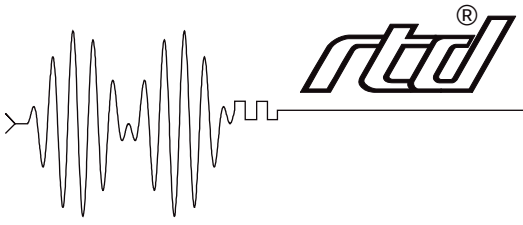
DM5804/DM6804
User's Manual



Real Time Devices USA, Inc.

"Accessing the Analog World"™

Publication No. 5804-6/18/99

DM5804/DM6804
User's Manual

REAL TIME DEVICES USA, INC.
Post Office Box 906
State College, Pennsylvania 16804
Phone: (814) 234-8087
FAX: (814) 234-5218

Published by
Real Time Devices USA, Inc.
P.O. Box 906
State College, PA 16804

Copyright © 1992 by Real Time Devices, Inc.
All rights reserved

Printed in U.S.A.

Table of Contents

INTRODUCTION ***i*-1**

Am9513A Timer/Counter *i*-3
Digital I/O *i*-3
What Comes With Your Board *i*-3
Board Accessories *i*-3
Using This Manual *i*-3
When You Need Help *i*-4

CHAPTER 1 — BOARD SETTINGS **1-1**

Factory-Configured Switch and Jumper Settings 1-3
P3 — Interrupt Channel Select (Factory Setting: G Connected, Interrupt Channels Disabled) 1-4
P4 — Interrupt Source Select (Factory Setting: EXTINT) 1-4
S1 — Base Address (Factory Setting: 300 hex (768 decimal)) 1-5
Pull-up/Pull-down Resistors on Digital I/O Lines 1-6

CHAPTER 2 — BOARD INSTALLATION **2-1**

Board Installation 2-3
External I/O Connections 2-3
Connecting the Timer/Counters and Digital I/O 2-4
Running the 5804DIAG Diagnostics Program 2-4

CHAPTER 3 — HARDWARE DESCRIPTION **3-1**

Am9513A Timer/Counters 3-3
Digital I/O, Programmable Peripheral Interface 3-3
Interrupts 3-4

CHAPTER 4 — BOARD OPERATION AND PROGRAMMING **4-1**

Defining the I/O Map 4-3
BA + 0: PPI Port A — Digital I/O (Read/Write) 4-3
BA + 1: PPI Port B — Digital I/O (Read/Write) 4-3
BA + 2: PPI Port C — Digital I/O (Read/Write) 4-3
BA + 3: 8254 PPI Control Word (Write Only) 4-4
BA + 4: Am9513A Data Register (Read/Write) 4-5
BA + 5: Am9513A Command Register (Read/Write) 4-5
BA + 6: IRQ Enable (Write Only) 4-6
BA + 7: Interrupt Status/Clear (Read/Write) 4-6
Programming the DM5804 4-7
Clearing and Setting Bits in a Port 4-8
Initializing the Am9513A 4-9
Initializing the 8255 4-9
Digital I/O Operations 4-9
Interrupts 4-10
What Is an Interrupt? 4-10
Interrupt Request Lines 4-10
8259 Programmable Interrupt Controller 4-10
Interrupt Mask Register (IMR) 4-10
End-of-Interrupt (EOI) Command 4-10
What Exactly Happens When an Interrupt Occurs? 4-11

Using Interrupts in Your Programs 4-11
Writing an Interrupt Service Routine (ISR) 4-11
Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector 4-12
Restoring the Startup IMR and Interrupt Vector 4-13
Common Interrupt Mistakes 4-13
Example Programs 4-14
C and Pascal Programs 4-14
BASIC Programs 4-14

CHAPTER 5 — EXAMPLES OF Am9513A APPLICATIONS 5-1

EXAMPLE: Counting Program Using Timer/Counters 1, 2, and 3 5-3
EXAMPLE: Setting Up the Am9513A as a Frequency Counter 5-8

APPENDIX A — DM5804 SPECIFICATIONS A-1

APPENDIX B — P2 CONNECTOR PIN ASSIGNMENTS B-1

APPENDIX C — COMPONENT DATA SHEETS C-1

APPENDIX D — WARRANTY D-1

List of Illustrations

1-1 Board Layout Showing Factory-Configured Settings 1-3
1-2 Interrupt Channel Select Jumper, P3 1-4
1-3 Pulling Down the Interrupt Request Line 1-4
1-4 Interrupt Source Select Jumper, P4 1-5
1-5 Base Address Switch, S1 1-6
1-6 Port A and Port B Pull-up/Pull-down Resistor Circuitry 1-6
1-7 Adding Pull-ups and Pull-downs to Some Digital I/O Lines 1-7
2-1 P2 I/O Connector Pin Assignments 2-4
3-1 DM5804 Block Diagram 3-3
5-1 Master Mode Register Bit Assignments 5-6
5-2 Counter Mode Register Bit Assignments 5-7
5-3 Frequency Scaler Ratio 5-8

INTRODUCTION

The DM5804 dataModule™ timer/counter and digital I/O board turns your IBM PC-compatible cpuModule™ or other PC/104 computer into a high-performance timing, counting, and control system. Ultra-compact for embedded and portable applications, the DM5804 features:

- Five general purpose 16-bit timer/counters in an Am9513A chip,
- 24 timer/counter modes of operation,
- Binary or BCD up or down counting,
- On-board 5 MHz crystal,
- 24 TTL/CMOS 8255-based digital I/O lines which can be configured with pull-up or pull-down resistors,
- Operation from +5 volts only,
- BASIC, Turbo Pascal, and Turbo C source code; diagnostics program.

The following paragraphs briefly describe the major functions of the board. A more detailed discussion of board functions is included in Chapter 3, *Hardware Operation*, and Chapter 4, *Board Operation and Programming*. The board setup is described in Chapter 1, *Board Settings*.

Am9513A Timer/Counter

The versatile Am9513A general purpose timer/counter provides a variety of timing, sequencing, and counting functions. The Am9513A chip contains five 16-bit timer/counters which can be used individually or internally cascaded to form a counter of up to 80 bits. With 24 operating modes, up or down counting in binary or BCD, and hardware or software gating, these timer/counters can be easily tailored for a wide variety of applications. The timer/counters are clocked by an on-board 5 MHz crystal. The source, gate, and output for each timer/counter is available at the P2 I/O connector.

Digital I/O

The DM5804 has 24 TTL/CMOS-compatible digital I/O lines which can be directly interfaced with external devices or signals to sense switch closures, trigger digital events, or activate solid-state relays. These lines are provided by the on-board 8255 programmable peripheral interface chip. Pads for installing and activating pull-up or pull-down resistors are included on the board. Installation procedures are given at the end of Chapter 1, *Board Settings*.

AT Bus Connector J6 (DM6804)

The DM6804 is exactly the same as the DM5804 except for the addition of the AT bus connector J6. This allows you to stack the module with CPU's that have the AT bus connectors and access the AT interrupts.

What Comes With Your Board

You receive the following items in your DM5804 package:

- DM5804 interface board with stackthrough bus header
- Software and diagnostics diskette with BASIC, Turbo Pascal, and Turbo C source code
- User’s manual

If any item is missing or damaged, please call Real Time Devices’ Customer Service Department at (814) 234-8087. If you require service outside the U.S., contact your local distributor.

Board Accessories

In addition to the items included in your DM5804 package, Real Time Devices offers a full line of board accessories. Call your local distributor or our main office for more information about these accessories and for help in choosing the best items to support your board’s application. Accessories for the DM5804 include the TB50 terminal board and XB50 prototype/terminal board for prototype development and easy signal access, the DM14 extender board for testing your module and XT50 twisted pair wire flat ribbon cable assembly for external interfacing.

Using This Manual

This manual is intended to help you install your new board and get it running quickly, while also providing enough detail about the board and its functions so that you can enjoy maximum use of its features even in the most complex applications. We assume that you already have an understanding of data acquisition and control principles and that you can customize the example software or write your own applications programs.

When You Need Help

This manual and the example programs in the software package included with your board provide enough information to properly use all of the board's features. If you have any problems installing or using this board, contact our Technical Support Department, (814) 234-8087, during regular business hours, eastern standard time or eastern daylight time, or send a FAX requesting assistance to (814) 234-5218. When sending a FAX request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

CHAPTER 1

BOARD SETTINGS

The DM5804 has jumper and switch settings you can change if necessary for your application. The board is factory-configured with the most often used settings. The factory settings are listed and shown on a diagram in the beginning of this chapter. Should you need to change these settings, use these easy-to-follow instructions before you install the board in your system.

Note that by installing resistor packs at four locations around the 8255 PPI and soldering jumpers in the desired locations on the associated pads, you can configure your 8255 digital I/O lines to be pulled up or pulled down. This procedure is explained at the end of this chapter.

Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumper and switches on the DM5804. Figure 1-1 shows the board layout and the locations of the factory-set jumpers. The following paragraphs explain how to change the factory settings. Pay special attention to the setting of S1, the base address switch, to avoid address contention when you first use your board in your system.

Table 1-1 Factory Settings

Switch/ Jumper	Function Controlled	Factory Settings (Jumpers Installed)
P3	Connects one of the four sources selected on P4 to an interrupt channel; pulls tri-state buffer to ground (G) for multiple interrupt applications	G (ground for buffer); interrupt channels disabled
P4	Selects the interrupt source	EXTINT
S1	Sets the base address	300 hex (768 decimal)

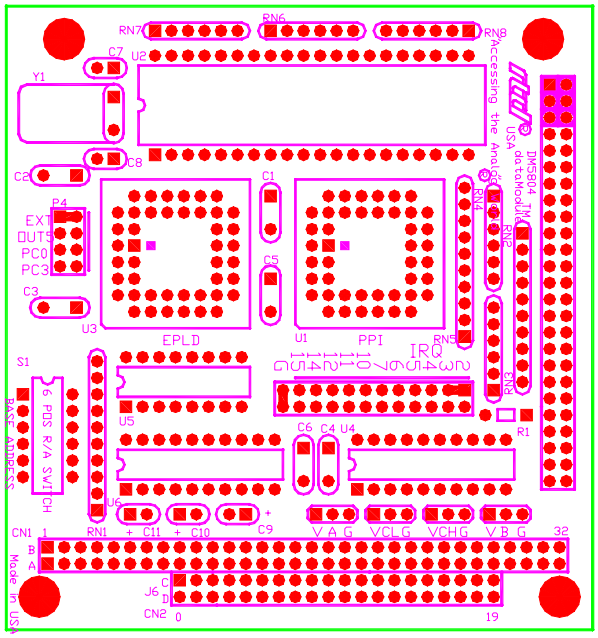


Fig. 1-1 — Board Layout Showing Factory-Configured Settings

P3 — Interrupt Channel Select (Factory Setting: G Connected, Interrupt Channels Disabled)

This header connector, shown in Figure 1-2, lets you connect an interrupt source selected on P4 to an interrupt channel, IRQ2 through IRQ7 . IRQ10, 11, 12, 14 and 15 can only be used if you have the DM6804 with the AT connector J6 installed. To connect the interrupt source selected on P4 to an IRQ channel, you must install a jumper across the desired IRQ channel. Figure 1-2a shows the factory setting and Figure 1-2b shows IRQ3 selected.

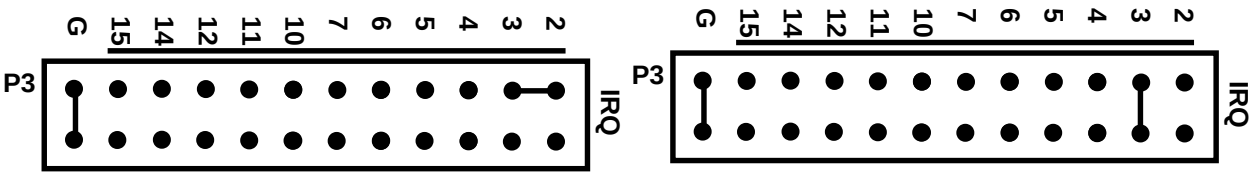


Fig. 1-2a — IRQ Disabled

Fig. 1-2b — IRQ3 Selected

Fig. 1-2 — Interrupt Channel Select Jumper, P3

The leftmost pair of pins on P3, labeled G, are provided so that you can install a jumper which connects a 1 kilohm pull-down resistor to the output of a high-impedance tri-state driver which carries the interrupt request signal. This pull-down resistor drives the interrupt request line low whenever interrupts are not active. So, whenever an interrupt request is made, the tri-state buffer is enabled, forcing the output high and causing an interrupt. You can monitor the interrupt status through bit 0 in the status word (I/O address location BA + 7). After the interrupt has been serviced, the clear command returns the IRQ line low, disabling the tri-state buffers, and pulling the output low again. Figure 1-3 shows this circuit. Because the interrupt request line is driven low only by the pull-down resistor, you can have two or more boards which share the same IRQ channel. You can tell which board issued the interrupt request by monitoring each board's IRQ status bit.

NOTE: When you use multiple boards that share the same interrupt, only one board should have the G jumper installed. The rest should be disconnected. Whenever you operate a single board, the G jumper should be installed.

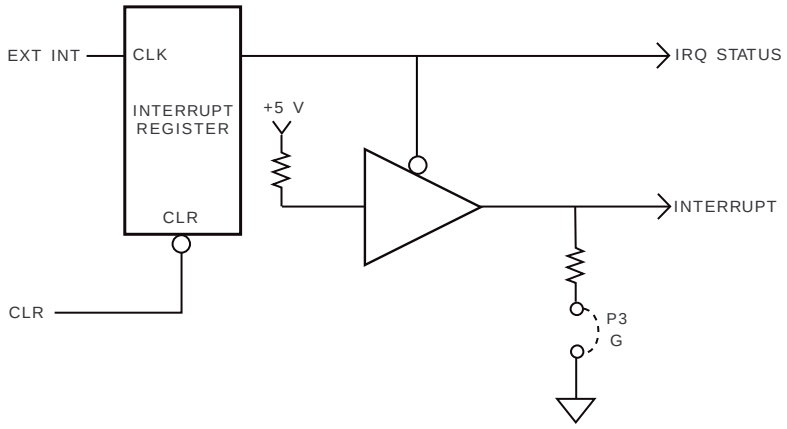


Fig. 1-3 — Pulling Down the Interrupt Request Line

P4 — Interrupt Source Select (Factory Setting: EXTINT)

This header connector, shown in Figure 1-4, lets you connect one of four interrupt sources for interrupt generation. These sources are: PC3, which is the INTRA signal from the 8255 PPI; PC0, which is the INTRB signal from the 8255 PPI; OUT5, the output from Am9513A timer/counter 5; and EXTINT, an external interrupt you can route onto the board through the P2 I/O connector. To connect an interrupt source, place the jumper across the desired set of pins. Note that only ONE interrupt source can be activated at a time.

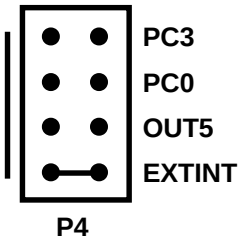


Fig. 1-4 — Interrupt Source Select Jumper, P4

S1 — Base Address (Factory Setting: 300 hex (768 decimal))

One of the most common causes of failure when you are first trying your board is address contention. Some of your computer's I/O space is already occupied by internal I/O and other peripherals. When the DM5804 board attempts to use I/O address locations already used by another device, contention results and the board does not work.

To avoid this problem, the DM5804 has an easily accessible DIP switch, S1, which lets you select any one of 64 starting addresses in the computer's I/O. Should the factory setting of 300 hex (768 decimal) be unsuitable for your system, you can select a different base address simply by setting the switches to any value shown in Table 1-2. The table shows the switch settings and their corresponding decimal and hexadecimal (in parentheses) values. Make sure that you verify the order of the switch numbers on the switch (1 through 6) before setting them. When the switches are pulled forward, they are OPEN, or set to logic 1, as labeled on the DIP switch package. When you set the base address for your board, record the value in the table inside the back cover. Figure 1-5 shows the DIP switch set for a base address of 300 hex (768 decimal).

Table 1-2 Base Address Switch Settings, S1

Base Address Decimal / (Hex)	Switch Setting 6 5 4 3 2 1	Base Address Decimal / (Hex)	Switch Setting 6 5 4 3 2 1	Base Address Decimal / (Hex)	Switch Setting 6 5 4 3 2 1	Base Address Decimal / (Hex)	Switch Setting 6 5 4 3 2 1
512 / (200)	0 0 0 0 0 0	640 / (280)	0 1 0 0 0 0	768 / (300)	1 0 0 0 0 0	896 / (380)	1 1 0 0 0 0
520 / (208)	0 0 0 0 0 1	648 / (288)	0 1 0 0 0 1	776 / (308)	1 0 0 0 0 1	904 / (388)	1 1 0 0 0 1
528 / (210)	0 0 0 0 1 0	656 / (290)	0 1 0 0 1 0	784 / (310)	1 0 0 0 1 0	912 / (390)	1 1 0 0 1 0
536 / (218)	0 0 0 0 1 1	664 / (298)	0 1 0 0 1 1	792 / (318)	1 0 0 0 1 1	920 / (398)	1 1 0 0 1 1
544 / (220)	0 0 0 1 0 0	672 / (2A0)	0 1 0 1 0 0	800 / (320)	1 0 0 1 0 0	928 / (3A0)	1 1 0 1 0 0
552 / (228)	0 0 0 1 0 1	680 / (2A8)	0 1 0 1 0 1	808 / (328)	1 0 0 1 0 1	936 / (3A8)	1 1 0 1 0 1
560 / (230)	0 0 0 1 1 0	688 / (2B0)	0 1 0 1 1 0	816 / (330)	1 0 0 1 1 0	944 / (3B0)	1 1 0 1 1 0
568 / (238)	0 0 0 1 1 1	696 / (2B8)	0 1 0 1 1 1	824 / (338)	1 0 0 1 1 1	952 / (3B8)	1 1 0 1 1 1
576 / (240)	0 0 1 0 0 0	704 / (2C0)	0 1 1 0 0 0	832 / (340)	1 0 1 0 0 0	960 / (3C0)	1 1 1 0 0 0
584 / (248)	0 0 1 0 0 1	712 / (2C8)	0 1 1 0 0 1	840 / (348)	1 0 1 0 0 1	968 / (3C8)	1 1 1 0 0 1
592 / (250)	0 0 1 0 1 0	720 / (2D0)	0 1 1 0 1 0	848 / (350)	1 0 1 0 1 0	976 / (3D0)	1 1 1 0 1 0
600 / (258)	0 0 1 0 1 1	728 / (2D8)	0 1 1 0 1 1	856 / (358)	1 0 1 0 1 1	984 / (3D8)	1 1 1 0 1 1
608 / (260)	0 0 1 1 0 0	736 / (2E0)	0 1 1 1 0 0	864 / (360)	1 0 1 1 0 0	992 / (3E0)	1 1 1 1 0 0
616 / (268)	0 0 1 1 0 1	744 / (2E8)	0 1 1 1 0 1	872 / (368)	1 0 1 1 0 1	1000 / (3E8)	1 1 1 1 0 1
624 / (270)	0 0 1 1 1 0	752 / (2F0)	0 1 1 1 1 0	880 / (370)	1 0 1 1 1 0	1008 / (3F0)	1 1 1 1 1 0
632 / (278)	0 0 1 1 1 1	760 / (2F8)	0 1 1 1 1 1	888 / (378)	1 0 1 1 1 1	1016 / (3F8)	1 1 1 1 1 1

0 = closed, 1 = open

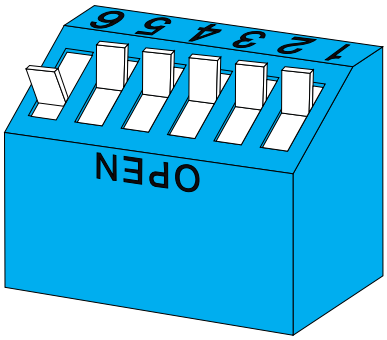


Fig. 1-5 — Base Address Switch, S1

Pull-up/Pull-down Resistors on Digital I/O Lines

The 8255 programmable peripheral interface provides 24 parallel TTL/CMOS compatible digital I/O lines which can be interfaced with external devices. The lines are divided into four groups: eight Port A lines, four Port C Lower lines, eight Port B lines, and four Port C Upper lines. You can install and connect pull-up or pull-down resistors for any or all of these four groups of lines. You may want to pull lines up for connection to switches. This will pull the line high when the switch is disconnected. Or, you may want to pull down lines connected to relays which control turning motors on and off. These motors turn on when the digital lines controlling them are high.

To use the pull-up/pull-down feature, you must first install 10 kilohm resistor packs in any or all of the four locations around the 8255, labeled PA, PB, PCL, and PCH. PA and PB take 10-pin packs, and CL and CH take 6-pin packs. Figure 1-6 shows a blowup of PA and PB.

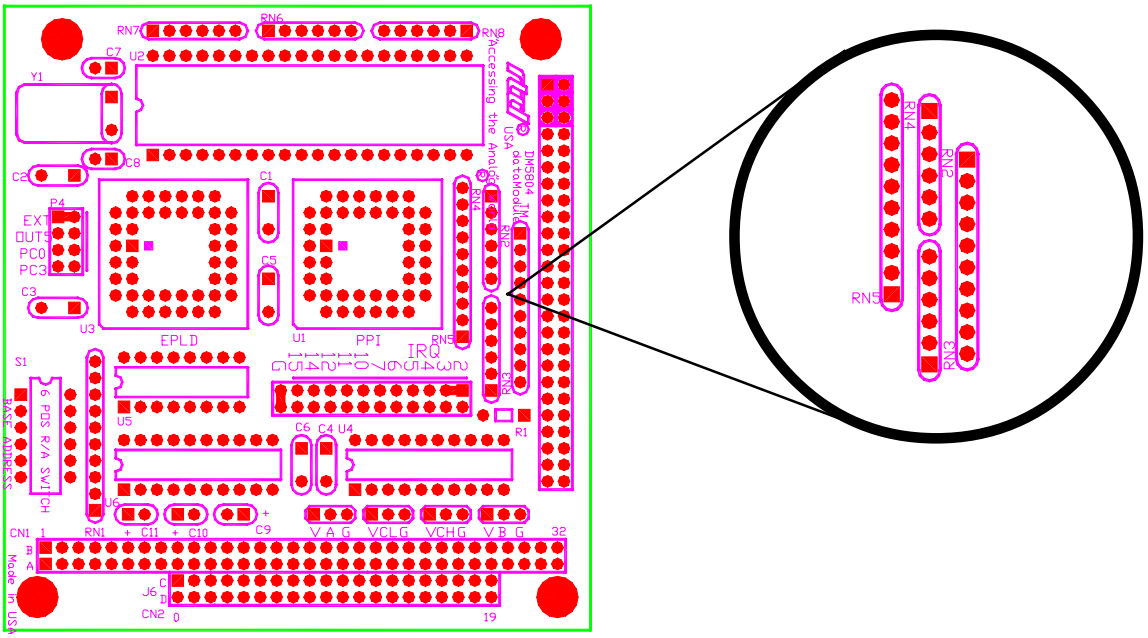


Fig. 1-6 — Port A and Port B Pull-up/Pull-down Resistor Circuitry

After the resistor packs are installed, you must connect them into the circuit as pull-ups or pull-downs. Locate the three-hole pads on the board near the resistor packs. They are labeled G (for ground) on one end and V (for Vcc) on the other end. The middle hole is common. PA is for Port A, PB for Port B, CL is for Port C Lower, and CH is for Port C Upper. Figure 1-6 shows a blowup of the pads for Port A and Port B. To operate as pull-ups, solder a jumper wire between the common pin (middle pin of the three) and the V pin. For pull-downs, solder a jumper wire between the common pin (middle pin) and the G pin. For example, Figure 1-7 shows Port A lines with pull-ups, Port C Lower with pull-downs, and Port C Upper with no resistors.

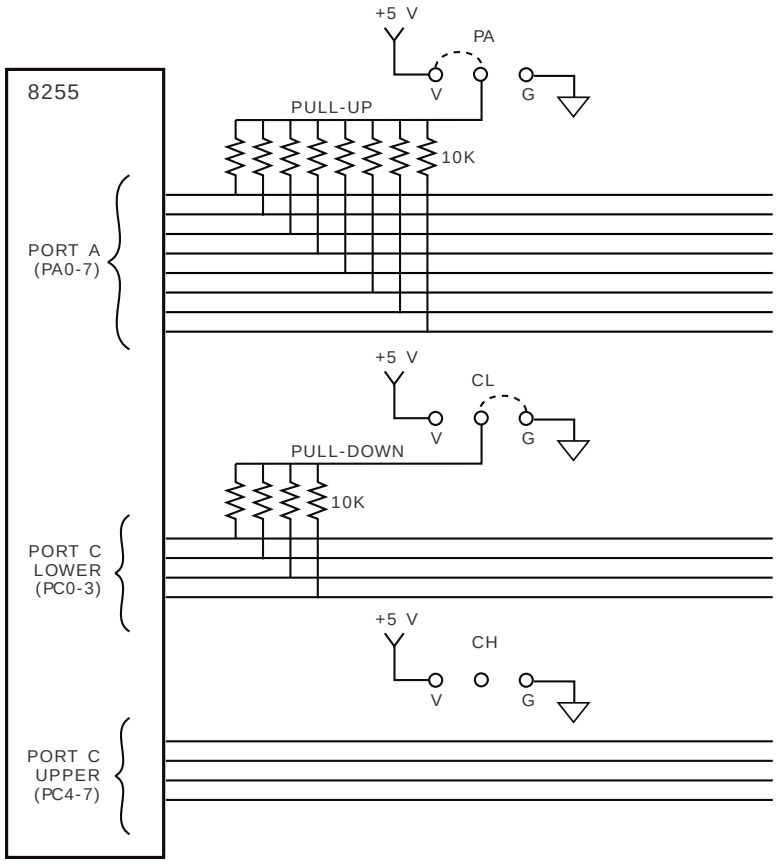


Fig. 1-7 — Adding Pull-ups and Pull-downs to Some Digital I/O Lines

CHAPTER 2

BOARD INSTALLATION

The DM5804 is easy to install in your cpuModule™ or other PC/104 based system. This chapter tells you step-by-step how to install and connect the board.

After you have installed the board and made all of your connections, you can turn your system on and run the 5804DIAG board diagnostics program included on your example software disk to verify that your board is working.

Board Installation

Keep the board in its antistatic bag until you are ready to install it in your cpuModule™ or other PC/104 based system. When removing it from the bag, hold the board at the edges and do not touch the components or connectors.

Before installing the board in your system, check the jumper and switch settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable board operation and erratic response.

The DM5804 comes with a stackthrough P1 connector. The stackthrough connector lets you stack another board on top of your DM5804, plugging it into the data bus through the pins on the non-component side of the board.

To install the board, follow the procedures described in the computer manual and the steps below:

1. Turn OFF the power to your system.
2. Touch the metal rack to discharge any static buildup and then remove the board from its antistatic bag.
3. Select the appropriate standoffs for your application to secure the board when you install it in your system (two sizes are included with the board).
4. Holding the board by its edges, orient it so that the P1 bus connector's pin 1 lines up with pin 1 of the expansion connector onto which you are installing the board.
5. After carefully positioning the board so that the DM5804's bus connector is resting on the expansion connector, gently and evenly press down on the board until it is secured on the connector.

   NOTE: Do not force the board onto the connector. If the board does not slide into place, remove it and try again. Wiggling the board or exerting too much pressure can result in damage to the DM5804 or to the module it is being stacked with.
6. After the board is installed, connect the cable to I/O connector P2 on the board. When making this connection, note that there is no keying to guide you in orientation. You must make sure that pin 1 of the cable is connected to pin 1 of P2 (pin 1 is marked on the board with a small square). For twisted pair cables, pin 1 is the dark brown wire; for standard single wire cables, pin 1 is the red wire.
7. Make sure all connections are secure.

External I/O Connections

Figure 2-1 shows the DM5804's P2 I/O connector pinout. Refer to this diagram as you make your I/O connections.

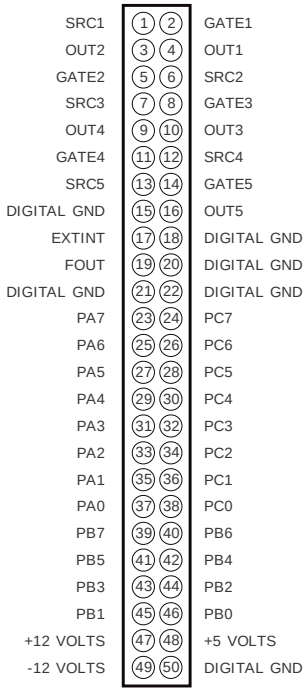

SRC1 1 2 GATE1
OUT2 3 4 OUT1
GATE2 5 6 SRC2
SRC3 7 8 GATE3
OUT4 9 10 OUT3
GATE4 11 12 SRC4
SRC5 13 14 GATE5
DIGITAL GND 15 16 OUT5
EXTINT 17 18 DIGITAL GND
FOUT 19 20 DIGITAL GND
DIGITAL GND 21 22 DIGITAL GND
PA7 23 24 PC7
PA6 25 26 PC6
PA5 27 28 PC5
PA4 29 30 PC4
PA3 31 32 PC3
PA2 33 34 PC2
PA1 35 36 PC1
PA0 37 38 PC0
PB7 39 40 PB6
PB5 41 42 PB4
PB3 43 44 PB2
PB1 45 46 PB0
+12 VOLTS 47 48 +5 VOLTS
-12 VOLTS 49 50 DIGITAL GND

Fig. 2-1 — P2 I/O Connector Pin Assignments

Connecting the Timer/Counters and Digital I/O

For all of the digital connections, the high side of an external signal source or destination device is connected to the appropriate signal pin on the I/O connector, and the low side is connected to any DIGITAL GND.

Running the 5804DIAG Diagnostics Program

Now that your board is ready to use, you will want to try it out. An easy-to-use, menu-driven diagnostics program, 5804DIAG, is included with your example software to help you verify your board's operation. You can also use this program to make sure that your current base address setting does not contend with another device.

CHAPTER 3

HARDWARE DESCRIPTION

This chapter describes the features of the DM5804 hardware. The major circuits are the timer/counters and the digital I/O lines. This chapter also describes the hardware-selectable interrupts.

The DM5804 has two major circuits, the timer/counters and the digital I/O lines. Figure 3-1 shows the block diagram of the board. This chapter describes the hardware which makes up the major circuits and hardware-selectable interrupts.

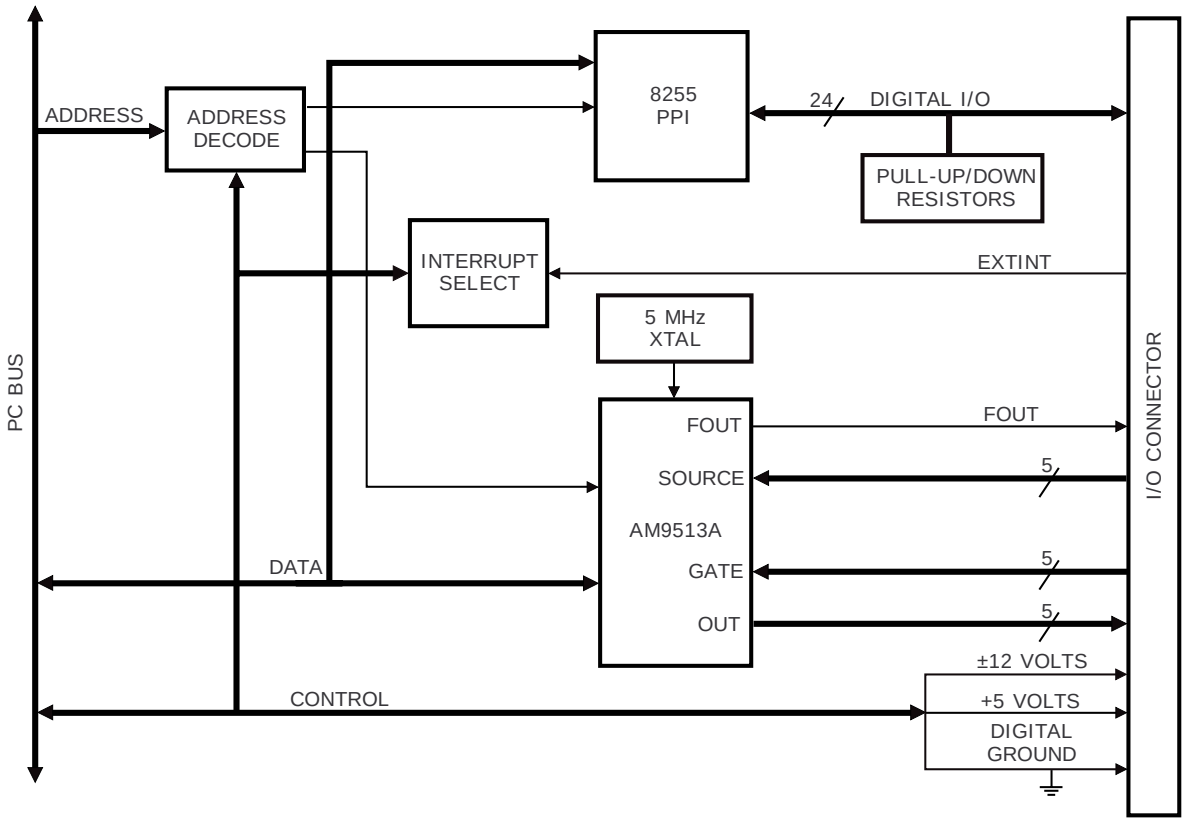


Fig. 3-1 — DM804 Block Diagram

Am9513A Timer/Counters

The Am9513A System Timing Controller contains five general purpose 16-bit timer/counters which are capable of performing many different types of counting, sequencing, and timing functions. The Am9513A supports up or down counting in binary or BCD with hardware or software gating of each counter. Its 24 modes of operation are detailed in the Am9513A Data Sheet reprint from AMD included in Appendix C.

The Am9513A is structured with a series of internal registers that set the mode of operation for each counter. These registers are fully described in Appendix C.

Any of the counters can be internally cascaded to create a counter of up to 80 bits. For example, two cascaded counters form a 32-bit counter for longer counting capability. Rarely is it practical to cascade more than three counters. Cascading is described in Appendix C, Chapter 3 of the Am9513A data sheet.

The timer/counters are driven by an on-board 5 MHz crystal oscillator.

Digital I/O, Programmable Peripheral Interface

The programmable peripheral interface (PPI) is used for digital I/O functions. This high-performance TTL/CMOS compatible chip has 24 digital I/O lines divided into two groups of 12 lines each:

Group A — Port A (8 lines) and Port C Upper (4 lines);
Group B — Port B (8 lines) and Port C Lower (4 lines).

All three ports, A, B, and C, are available at the I/O connector, P2. You can use these ports in one of these three PPI operating modes:

Mode 0 — Basic input/output. Lets you use simple input and output operation for a port. Data is written to or read from the specified port.

Mode 1 — Strobed input/output. Lets you transfer I/O data from Port A or Port B in conjunction with strobes or handshaking signals.

Mode 2 — Strobed bidirectional input/output. Lets you communicate bidirectionally with an external device through Port A. Handshaking is similar to Mode 1.

These modes are detailed in the 8255 Data Sheet, reprinted from Intel in Appendix C.

Interrupts

The DM5804 has four jumper-selectable interrupt sources: PC3, which is the INTRA signal from the 8255 PPI; PC0, which is the INTRB signal from the 8255 PPI; OUT5, the output from Am9513A timer/counter 5; and EXTINT, an external interrupt you can route onto the board through I/O connector P2. Chapter 1 tells you how to set the jumpers on the interrupt header connectors, P3 and P4, and Chapter 4 describes how to program interrupts.

CHAPTER 4

BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program your DM5804. It provides a complete description of the I/O map and a description of programming operations to aid you in programming. The example programs included on the disk in your board package are listed at the end of this chapter. These programs, written in Turbo C, Turbo Pascal, and BASIC, include source code to simplify your applications programming. Chapter 5 contains examples for setting up the timer/counters for specific applications.

Defining the I/O Map

The I/O map for the DM5804 is shown in Table 4-1 below. As shown, the board occupies eight consecutive I/O port locations. The base address (designated as BA) can be selected using DIP switch S1 as described in Chapter 1, *Board Settings*. This switch can be accessed without removing the board from the connector. The following sections describe the register contents of each address used in the I/O map.

Table 4-1: DM5804 I/O Map

Register Description	Read Function	Write Function	Address * (Decimal)
8255 PPI Port A	Read Port A digital input lines	Program Port A digital output lines	BA + 0
8255 PPI Port B	Read Port B digital input lines	Program Port B digital output lines	BA + 1
8255 PPI Port C	Read Port C digital input lines	Program Port C digital output lines	BA + 2
8255 PPI Control Word	Reserved	Program PPI configuration	BA + 3
Am9513A Data Word	Read data register	Program data register	BA + 4
Am9513A Control Word	Read control register	Program control register	BA + 5
IRQ Enable	Reserved	Enable and disable interrupt generation	BA + 6
Interrupt Status/Clear	Read status of interrupt	Clear interrupt	BA + 7
* BA = Base Address			

BA + 0: PPI Port A — Digital I/O (Read/Write)

Transfers the 8-bit Port A digital input and digital output data between the board and an external device. A read transfers data from the external device, through P2, and into PPI Port A; a write transfers the written data from Port A through P2 to an external device.

BA + 1: PPI Port B — Digital I/O (Read/Write)

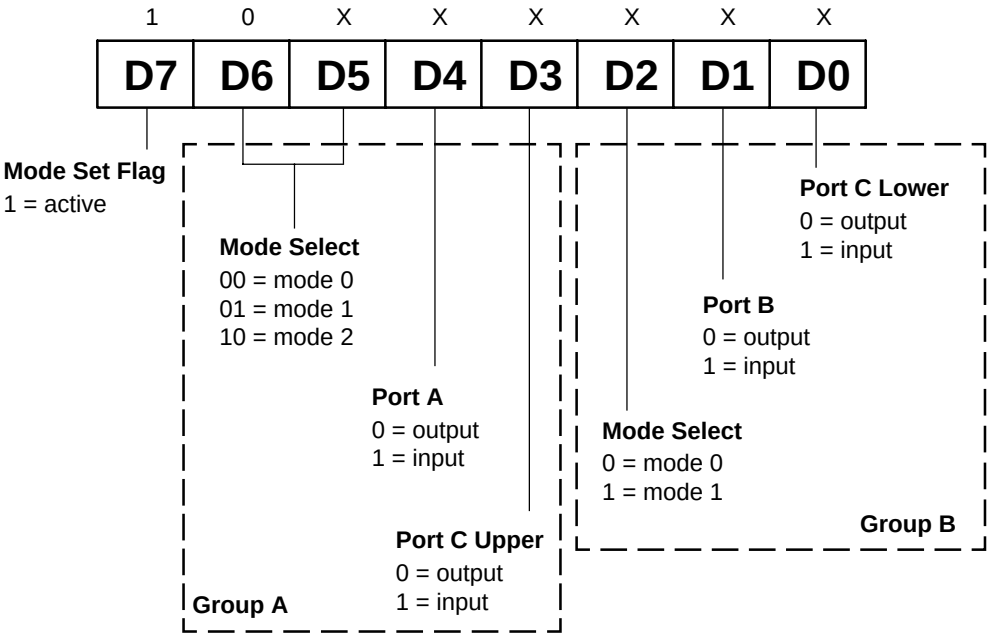
Transfers the 8-bit Port B digital input and digital output data between the board and an external device. A read transfers data from the external device, through P2, and into PPI Port B; a write transfers the written data from Port B through P2 to an external device.

BA + 2: PPI Port C — Digital I/O (Read/Write)

Transfers the two 4-bit Port C digital input and digital output data groups (Port C Upper and Port C Lower) between the board and an external device. A read transfers data from the external device, through P2, and into PPI Port C; a write transfers the written data from Port C through P2 to an external device.

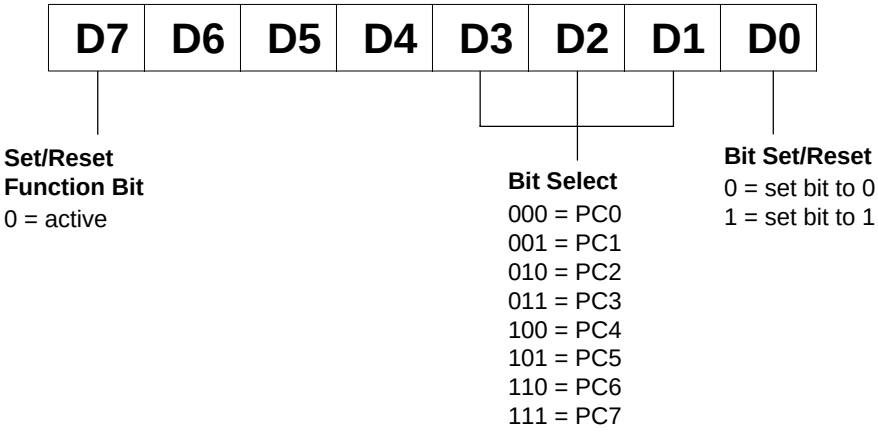
BA + 3: 8255 PPI Control Word (Write Only)

When bit 7 of this word is set to 1, a write programs the PPI configuration. The table below shows the control words for the 16 possible Mode 0 Port I/O combinations.

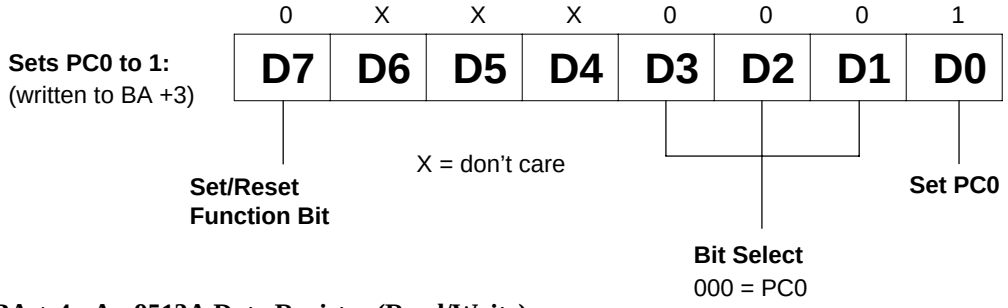


8255 Port I/O Flow Direction and Control Words, Mode 0						
Group A		Group B		Control Word		
Port A	Port C Upper	Port B	Port C Lower	Binary	Decimal	Hex
Output	Output	Output	Output	1 0 0 0 0 0 0 0	128	80
Output	Output	Output	Input	1 0 0 0 0 0 0 1	129	81
Output	Output	Input	Output	1 0 0 0 0 0 1 0	130	82
Output	Output	Input	Input	1 0 0 0 0 0 1 1	131	83
Output	Input	Output	Output	1 0 0 0 1 0 0 0	136	88
Output	Input	Output	Input	1 0 0 0 1 0 0 1	137	89
Output	Input	Input	Output	1 0 0 0 1 0 1 0	138	8A
Output	Input	Input	Input	1 0 0 0 1 0 1 1	139	8B
Input	Output	Output	Output	1 0 0 1 0 0 0 0	144	90
Input	Output	Output	Input	1 0 0 1 0 0 0 1	145	91
Input	Output	Input	Output	1 0 0 1 0 0 1 0	146	92
Input	Output	Input	Input	1 0 0 1 0 0 1 1	147	93
Input	Input	Output	Output	1 0 0 1 1 0 0 0	152	98
Input	Input	Output	Input	1 0 0 1 1 0 0 1	153	99
Input	Input	Input	Output	1 0 0 1 1 0 1 0	154	9A
Input	Input	Input	Input	1 0 0 1 1 0 1 1	155	9B

When bit 7 of this word is set to 0, a write can be used to individually program the Port C lines.



For example, if you want to set Port C bit 0 to 1, you would set up the control word so that bit 7 is 0; bits 1, 2, and 3 are 0 (this selects PC0); and bit 0 is 1 (this sets PC0 to 1). The control word is set up like this:



BA + 4: Am9513A Data Register (Read/Write)

Accesses the Am9513A data register. See data sheet included in Appendix C for more information on the operation of the Am9513A.

IMPORTANT

Because of the bus release time of the Am9513A, AMD recommends you insert a small delay between software accesses to the chip.

BA + 5: Am9513A Command Register (Read/Write)

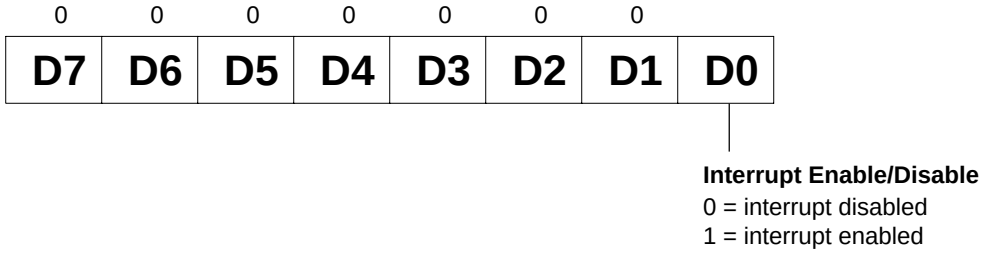
Accesses the Am9513A command register. See data sheet included in Appendix C for more information on the operation of the Am9513A.

IMPORTANT

Because of the bus release time of the Am9513A, AMD recommends you insert a small delay between software accesses to the chip.

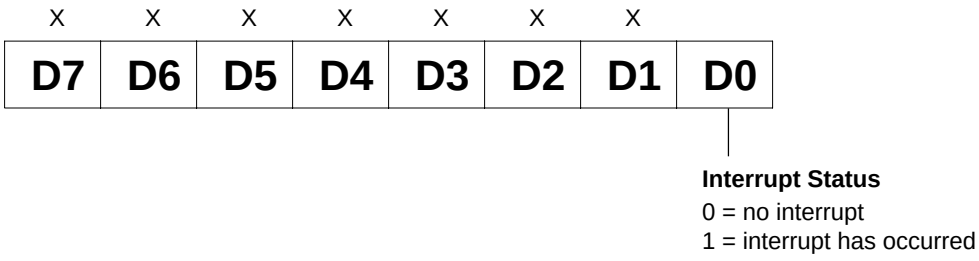
BA + 6: IRQ Enable (Write Only)

Enables and disables interrupt generation. Writing a “1” enables interrupt generation; writing a “0” disables interrupt generation.



BA + 7: Interrupt Status/Clear (Read/Write)

A read shows the status of the interrupt (bit 0 only) as defined below. A write clears the interrupt (data written is irrelevant). Each time the interrupt status bit goes high, a write should follow to clear the bit.



Programming the DM5804

This section gives you some general information about programming and the DM5804 board. Chapter 5 provides some specific programming examples, and the Am9513A data sheet in Appendix C provides detailed programming information for all 24 operating modes of the Am9513A. These tools will help you as you use the example programs included with the board. All of the program descriptions in this section use decimal values unless otherwise specified.

The DM5804 is programmed by writing to and reading from the correct I/O port locations on the board. These I/O ports were defined in the previous section. Most high-level languages such as BASIC, Pascal, C, and C++, and of course assembly language, make it very easy to read/write these ports. The table below shows you how to read from and write to I/O ports using some popular programming languages.

Language	Read	Write
BASIC	Data=INP(Address)	OUT Address,Data
Turbo C	Data=inportb(Address)	outportb(Address,Data)
Turbo Pascal	Data:=Port[Address]	Port[Address]:=Data

Assembly	mov dx,Address
	in al,dx
		mov dx,Address
		mov al,Data
		out dx,al

In addition to being able to read/write the I/O ports on the DM5804, you must be able to perform a variety of operations that you might not normally use in your programming. The table below shows you some of the operators discussed in this section, with an example of how each is used with Pascal, C, and BASIC. Note that the modulus operator is used to retrieve the least significant byte (LSB) of a two-byte word, and the integer division operator is used to retrieve the most significant byte (MSB).

Language	Modulus	Integer Division	AND	OR
C	% a = b % c	/ a = b / c	& a = b & c	| a = b | c
Pascal	MOD a := b MOD c	DIV a := b DIV c	AND a := b AND c	OR a := b OR c
BASIC	MOD a = b MOD c	\ a = b \ c	AND a = b AND c	OR a = b OR c

Many compilers have functions that can read/write either 8 or 16 bits from/to an I/O port. For example, Turbo Pascal uses **Port** for 8-bit port operations and **PortW** for 16 bits, Turbo C uses **inportb** for an 8-bit read of a port and **inport** for a 16-bit read. **Be sure to use only 8-bit operations with the DM5804!**

Clearing and Setting Bits in a Port

When you clear or set one or more bits in a port, you must be careful that you do not change the status of the other bits. You can preserve the status of all bits you do not wish to change by proper use of the AND and OR binary operators. Using AND and OR, single or multiple bits can be easily cleared in one operation.

To **clear** a single bit in a port, AND the current value of the port with the value b, where $b = 255 - 2^{bit}$.

Example: Clear bit 5 in a port. Read in the current value of the port, AND it with 223 ($223 = 255 - 2^5$), and then write the resulting value to the port. In BASIC, this is programmed as:

```
V = INP(PortAddress)
V = V AND 223
OUT PortAddress, V
```

To **set** a single bit in a port, OR the current value of the port with the value b, where $b = 2^{bit}$.

Example: Set bit 3 in a port. Read in the current value of the port, OR it with 8 ($8 = 2^3$), and then write the resulting value to the port. In Pascal, this is programmed as:

```
V := Port[PortAddress];
V := V OR 8;
Port[PortAddress] := V;
```

Setting or clearing more than one bit at a time is accomplished just as easily. To **clear** multiple bits in a port, AND the current value of the port with the value b, where b = 255 - (the sum of the values of the bits to be cleared). Note that the bits do not have to be consecutive.

Example: Clear bits 2, 4, and 6 in a port. Read in the current value of the port, AND it with 171 ($171 = 255 - 2^2 - 2^4 - 2^6$), and then write the resulting value to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 171;
outportb(port_address, v);
```

To **set** multiple bits in a port, OR the current value of the port with the value b, where b = the sum of the individual bits to be set. Note that the bits to be set do not have to be consecutive.

Example: Set bits 3, 5, and 7 in a port. Read in the current value of the port, OR it with 168 ($168 = 2^3 + 2^5 + 2^7$), and then write the resulting value back to the port. In assembly language, this is programmed as:

```
mov dx, PortAddress
in al, dx
or al, 168
out dx, al
```

Often, assigning a range of bits is a mixture of setting and clearing operations. You can set or clear each bit individually or use a faster method of first clearing all the bits in the range then setting only those bits that must be set using the method shown above for setting multiple bits in a port. The following example shows how this two-step operation is done.

Example: Assign bits 3, 4, and 5 in a port to 101 (bits 3 and 5 set, bit 4 cleared). First, read in the port and clear bits 3, 4, and 5 by ANDing them with 199. Then set bits 3 and 5 by ORing them with 40, and finally write the resulting value back to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 199;
v = v | 40;
outportb(port_address, v);
```

A final note: Don't be intimidated by the binary operators AND and OR and try to use operators for which you have a better intuition. For instance, if you are tempted to use addition and subtraction to set and clear bits in place of the methods shown above, DON'T! Addition and subtraction may seem logical, but they **will not work** if you try to clear a bit that is already clear or set a bit that is already set. For example, you might think that to set bit 5 of a port, you simply need to read in the port, add 32 (2^5) to that value, and then write the resulting value back to the port. This works fine if bit 5 is not already set. But, what happens when bit 5 *is* already set? Bits 0 to 4 will be unaffected and we can't say for sure what happens to bits 6 and 7, but we can say for sure that bit 5 ends up cleared instead of being set. A similar problem happens when you use subtraction to clear a bit in place of the method shown above.

Now that you know how to clear and set bits, we are ready to look at the programming steps for the DM5804 board functions.

Initializing the Am9513A

The Am9513A has a sophisticated internal architecture which is programmed through a series of internal registers. These internal registers are accessed by writing to and reading from only two I/O port locations: the Data Register port at BA + 4 and the Control Register port at BA + 5. In our example programs, we follow these steps to initialize the Am9513A:

1. Send a master reset to the Am9513A
2. Point to and set up the master mode register
3. Point to and set up counter 1 mode register
4. Point to counter 1 load register and load desired value
5. Point to and set up counter 2 mode register
6. Point to counter 2 load register and load desired value
   •
   •
11. Point to and set up counter 5 mode register
12. Point to counter 5 load register and load desired value
13. Load and arm counters

The examples on the disk and in Chapter 5 will aid you in programming the Am9513A for your application. These tools and the data sheet in Appendix C provide a comprehensive description of timer/counter operation.

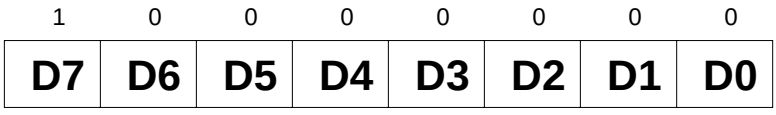
IMPORTANT

Because of the bus release time of the Am9513A, AMD recommends you insert a small delay between software accesses to the chip.

Initializing the 8255

Before you can use the digital I/O lines on your DM5804, the 8255 PPI must be initialized. This step must be executed every time you start up, reset, or reboot your computer.

The 8255 is initialized by writing the appropriate control word to I/O port BA + 3. The contents of your control word will vary, depending on how you want to configure your I/O lines. Use the control word description in the previous I/O map section to help you program the right value. In the example below, a decimal value of 128 sets up the 8255 so that all I/O lines are Mode 0 outputs.



Digital I/O Operations

Once the 8255 is initialized, you can use the digital I/O line to control or monitor external devices.

Interrupts

- What Is an Interrupt?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are very handy for dealing with asynchronous events (events that occur at less than regular intervals). Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time for it to do nothing while waiting for a keystroke to occur. Thus, the interrupt scheme is used and the processor proceeds with other tasks. Then, when a keystroke does occur, the keyboard 'interrupts' the processor, and the processor gets the keyboard data, places it in memory, and then returns to what it was doing before it was interrupted. Other common devices that use interrupts are modems, disk drives, and mice.

Your DM5804 board can interrupt the processor when one of the four interrupt sources is enabled through the jumper settings on P3 and P4. By using this interrupt, you can write software that effectively deals with real world events.

- Interrupt Request Lines

To allow different peripheral devices to generate interrupts on the same computer, the PC bus has eight different interrupt request (IRQ) lines. A transition from low to high on one of these lines generates an interrupt request which is handled by the PC's interrupt controller. The interrupt controller checks to see if interrupts are to be acknowledged from that IRQ and, if another interrupt is already in progress, it decides if the new request should supersede the one in progress or if it has to wait until the one in progress is done. This prioritizing allows an interrupt to be interrupted if the second request has a higher priority. The priority level is based on the number of the IRQ; IRQ0 has the highest priority, IRQ1 is second-highest, and so on through IRQ7, which has the lowest. Many of the IRQs are used by the standard system resources. IRQ0 is used by the system timer, IRQ1 is used by the key-board, IRQ3 by COM2, IRQ4 by COM1, and IRQ6 by the disk drives. Therefore, it is important for you to know which IRQ lines are available in your system for use by the DM5804 board.

- 8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in the PC is the 8259 Programmable Interrupt Controller. To use interrupts, you will need to know how to read and set the 8259's interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to the 8259.

- Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of an IRQ line; bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. If a bit is **set** (equal to 1), then the corresponding IRQ is masked and it will not generate an interrupt. If a bit is **clear** (equal to 0), then the corresponding IRQ is unmasked and can generate interrupts. The IMR is programmed through port 21H.

IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

I/O Port 21H

For all bits:
0 = IRQ unmasked (enabled)
1 = IRQ masked (disabled)

- End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 interrupt controller must be notified. This is done by writing the value 20H to I/O port 20H.

- What Exactly Happens When an Interrupt Occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to properly write software interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the DM5804), the interrupt controller checks to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed on the stack for storage, and a new CS and IP are loaded from a table that exists in the lowest 1024 bytes of memory. This table is referred to as the interrupt vector table and each entry is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor begins executing the code located at CS:IP. When the interrupt routine is completed, the CS, IP, and flags that were pushed on the stack when the interrupt occurred are now popped from the stack and execution resumes from the point where it was interrupted.

- Using Interrupts in Your Programs

Adding interrupts to your software is not as difficult as it may seem, and what they add in terms of performance is often worth the effort. Note, however, that although it is not that hard to use interrupts, the smallest mistake will often lead to a system hang that requires a reboot. This can be both frustrating and time-consuming. But, after a few tries, you'll get the bugs worked out and enjoy the benefits of properly executed interrupts. In addition to reading the following paragraphs, study the source code included on your DM5804 program disk in the interrupt programs for a better understanding of interrupt program development.

- Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write the interrupt service routine (ISR). This is the routine that will automatically be executed each time an interrupt request occurs on the specified IRQ. An ISR is different than standard routines that you write. First, on entrance, the processor registers should be pushed onto the stack **BEFORE** you do anything else. Second, just before exiting your ISR, you must clear the interrupt status of the DM5804 and write an end-of-interrupt command to the 8259 controller. Finally, when exiting the ISR, in addition to popping all the registers you pushed on entrance, you must use the IRET instruction and **not** a plain RET. The IRET automatically pops the flags, CS, and IP that were pushed when the interrupt was called.

If you find yourself intimidated by interrupt programming, take heart. Most Pascal and C compilers allow you to identify a procedure (function) as an interrupt type and will automatically add these instructions to your ISR, with one important exception: most compilers **do not** automatically add the end-of-interrupt command to the procedure; you must do this yourself. Other than this and the few exceptions discussed below, you can write your ISR just like any other routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend that you stick to the basics; just something that will convince you that it works, such as incrementing a global variable.

NOTE: If you are writing an ISR using assembly language, you are responsible for pushing and popping registers and using IRET instead of RET.

There are a few cautions you must consider when writing your ISR. The most important is, **do not use any DOS functions or routines that call DOS functions from within an ISR**. DOS is **not** reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about when using interrupts? Then, you could have a situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to DOS function X, then function X is essentially being called while it is already active. Such a reentrancy attempt spells disaster because DOS functions are not written to support it. This is a complex concept and you do not need to understand it. Just make sure that you do not call any DOS functions from within your ISR. The one wrinkle is that, unfortunately, it is not obvious which library routines included with your compiler use DOS functions. A rule of thumb is that routines which write to the screen, or check the status of or read the keyboard, and any disk I/O routines use DOS and should be avoided in your ISR.

The same problem of reentrancy exists for many floating point emulators as well, meaning you may have to avoid floating point (real) math in your ISR.

Note that the problem of reentrancy exists, no matter what programming language you are using. Even if you are writing your ISR in assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways around this problem, such as those which involve checking to see if any DOS functions are currently active when your ISR is called, but such solutions are well beyond the scope of this discussion.

The second major concern when writing your ISR is to make it as short as possible in terms of execution time. Spending long periods of time in your ISR may mean that other important interrupts are being ignored. Also, if you spend too long in your ISR, it may be called again before you have completed handling the first run. This often leads to a hang that requires a reboot.

Your ISR should have this structure:

- Push any processor registers used in your ISR. Most C and Pascal interrupt routines automatically do this for you.
- Put the body of your routine here.
- Clear the interrupt bit on the DM5804 by writing any value to BA + 7
- Issue the EOI command to the 8259 interrupt controller by writing 20H to port 20H.
- Pop all registers pushed on entrance. Most C and Pascal interrupt routines automatically do this for you.

The following C and Pascal examples show what the shell of your ISR should be like:

In C:

```
void interrupt ISR(void)
{
   /* Your code goes here. Do not use any DOS functions! */
   outportb(BaseAddress + 7, 0);       /* Clear DM5804 interrupt */
   outportb(0x20, 0x20);               /* Send EOI command to 8259 */
}
```

In Pascal:

```
Procedure ISR; Interrupt;
begin
   { Your code goes here. Do not use any DOS functions! }
   Port[BaseAddress + 7] := 0;      { Clear DM5804 interrupt }
   Port[$20] := $20;                { Send EOI command to 8259 }
end;
```

- Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector

The next step after writing the ISR is to save the startup state of the interrupt mask register and the interrupt vector that you will be using. The IMR is located at I/O port 21H. The interrupt vector you will be using is located in the interrupt vector table which is simply an array of 256-bit (4-byte) pointers and is located in the first 1024 bytes of memory (Segment = 0, Offset = 0). You can read this value directly, but it is a better practice to use DOS function 35H (get interrupt vector). Most C and Pascal compilers provide a library routine for reading the value of a vector. The vectors for the hardware interrupts are vectors 8 through 15, where IRQ0 uses vector 8, IRQ1 uses vector 9, and so on. Thus, if the DM5804 will be using IRQ3, you should save the value of interrupt vector 11.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read in the current IMR at I/O port 21H and **set** the bit that corresponds to your IRQ (remember, setting a bit disables interrupts on that IRQ while clearing a bit enables them). The IMR is arranged so that bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. See the paragraph entitled *Interrupt Mask Register (IMR)* earlier in this chapter for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21H.

With the startup IMR saved and the interrupts on your IRQ temporarily disabled, you can assign the interrupt vector to point to your ISR. Again, you can overwrite the appropriate entry in the vector table with a direct memory write, but this is a bad practice. Instead, use either DOS function 25H (set interrupt vector) or, if your compiler provides it, the library routine for setting an interrupt vector. Remember that vector 8 is for IRQ0, vector 9 is for IRQ1, and so on.

If you need to program the source of your interrupts, do that next. For example, if you are using the programmable interval timer to generate interrupts, you must program it to run in the proper mode and at the proper rate.

Finally, clear the bit in the IMR for the IRQ you are using. This enables interrupts on the IRQ.

– Restoring the Startup IMR and Interrupt Vector

Before exiting your program, you must restore the interrupt mask register and interrupt vectors to the state they were in when your program started. To restore the IMR, write the value that was saved when your program started to I/O port 21H. Restore the interrupt vector that was saved at startup with either DOS function 35H (get interrupt vector), or use the library routine supplied with your compiler. Performing these two steps will guarantee that the interrupt status of your computer is the same after running your program as it was before your program started running.

- Common Interrupt Mistakes

- Remember that hardware interrupts are numbered 8 through 15, even though the corresponding IRQs are numbered 0 through 7.
- Two of the most common mistakes when writing an ISR are forgetting to clear the interrupt status of the DM5804 and forgetting to issue the EOI command to the 8259 interrupt controller before exiting the ISR.

Example Programs

Included with the DM5804 is a set of example programs that demonstrate the use of many of the board's features. These examples are in written in C, Pascal, and BASIC. Also included is an easy-to-use menu-driven diagnostics program, 804DIAG, which is especially helpful when you are first checking out your board after installation.

C and Pascal Programs

These programs are source code files so that you can easily develop your own custom software for your DM5804.

Timer/Counter:

INTRPTS	Shows how to generate interrupts and read the digital I/O lines.
COUNT	Shows how to use the Am9513A as a simple counter.

Digital I/O:

DIGITAL	Simple program that shows how to read and write the digital I/O lines.

BASIC Programs

These programs are source code files so that you can easily develop your own custom software for your DM5804.

Timer/Counter:

COUNT	Shows how to use the Am9513A as a simple counter.
FCOUNT	Shows how to create a frequency counter using the Am9513A.

Digital I/O:

DIGITAL	Simple program that shows how to read and write the digital I/O lines.

CHAPTER 5

EXAMPLES OF Am9513A APPLICATIONS

This chapter steps through some example programs to help you understand how the Am9513A registers are programmed. The data pointer register and command registers are summarized in tables. The master mode and counter mode register bit assignments are also included, as well as the frequency scaler ratios.

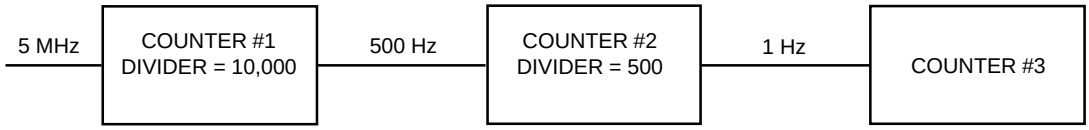
This chapter provides a more detailed look at some example programs using the Am9513A for counting and timing functions. If you are unfamiliar with the Am9513A and how it is programmed, walking through these examples and the other example programs included on your DM5804 disk may be the best way to understand the many registers and their operation so that you can successfully develop your own programs for your specific applications.

IMPORTANT

Because of the bus release time of the Am9513A, AMD recommends you insert a small delay between software accesses to the chip.

EXAMPLE: Counting Program Using Timer/Counters 1, 2, and 3

This BASIC program, COUNT on the example disk, shows you how to program the Am9513A's timer/counters 1, 2, and 3 to perform a simple counting function. In this example, counter 1 is used to divide the on-board 5 MHz clock by 10,000. The output from counter 1 (5 MHz ÷ 10,000 = 500 Hz) is used to clock counter 2. Counter 2 is used to divide this 500 Hz clock by 500. The result is a 1 Hz clock which is used to clock counter 3. Counter 3 counts the 1 Hz pulses. The count value from counter 3 is displayed on the screen. This value should start at 0 and increment once each second.



The first lines of the program clear the screen and set up the base address of the DM5804. The address in the variable "BA" must match the setting of the base address switch, S1, on the board. The factory setting of S1 is 300 hex (768 decimal).

```
CLS
INPUT "ENTER BASE ADDRESS IN DECIMAL: "; BA
```

The next section of the program sets up the computer screen:

```
CLS                              'CLEAR SCREEN
DIM RESULT AS LONG               'DIMENSION VARIABLE "RESULT" AS A LONG INTEGER
KEY(1) ON                        'TURN F1 KEY ON
LOCATE 2, 25
PRINT "DM5804 COUNTER DEMO PROGRAM !";
LOCATE 10, 31
PRINT "COUNTER #3 VALUE:";
LOCATE 24, 2
PRINT "F1 = QUIT ";
```

The next section of the program sets up the address for the DM5804 registers. These addresses are defined at the beginning of Chapter 4.

```
PA = BA + 0                      'ADDRESS FOR 8255 PORT A
PB = BA + 1                      'ADDRESS FOR 8255 PORT B
PC = BA + 2                      'ADDRESS FOR 8255 PORT C
CW = BA + 3                      'ADDRESS FOR 8255 CONTROL WORD
DR = BA + 4                      'ADDRESS FOR AM9513A COUNTER DATA REGISTER
CR = BA + 5                      'ADDRESS FOR AM9513A COUNTER CONTROL REGISTER
IRQEN = BA + 6                   'ADDRESS FOR INTERRUPT ENABLE
STAT = BA + 7                    'ADDRESS TO READ INTERRUPT STATUS
IRQCLR = BA + 7                  'ADDRESS TO CLEAR INTERRUPT STATUS BIT
```

Now, reset the Am9513A timer/counter chip (see Table 5-2):

```
OUT CR, &HFF                     'AM9513A MASTER RESET
```

Table 5-1 Load Data Pointer Commands				
	Element Cycle			Hold Cycle
	Mode Register	Load Register	Hold Register	Hold Register
Counter 1	01	09	11	19
Counter 2	02	0A	12	1A
Counter 3	03	0B	13	1B
Counter 4	04	0C	14	1C
Counter 5	05	0D	15	1D
Master Mode Register = 17 Alarm 1 Register = 07 Alarm 2 Register = 0F Status Register = 1F				
NOTE: All codes are in hex.				

Next, set up the Am9513A master mode register (see Figure 5-1). These are the settings we will use:

Scaler Control = binary division
Data Pointer Control = disable increment
Data Bus Width = 8 bits
FOUT Gate = FOUT on
FOUT Divider = divide by 16
FOUT Source = F1 (see Figure 5-3)
Compare 2 Enable = disabled
Compare 1 Enable = disabled
Time-of-Day Mode = disabled

VALUE = HEX 4000

```
OUT CR, &H17                    'POINT TO MASTER MODE REGISTER (TABLE 5-1)
OUT DR, &H0                     'MASTER MODE LSB
OUT DR, &H40                    'MASTER MODE MSB
```

Next, set up the counter 1 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = F1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count down
Output Control = TC toggled

VALUE = HEX 0B22

```
OUT CR, &H1                     'POINT TO COUNTER 1 MODE REGISTER (TABLE 5-1)
OUT DR, &H22                    'COUNTER 1 MODE LSB
OUT DR, &H0B                    'COUNTER 1 MODE MSB
```

Table 5-2 Am9513A Command Summary								
Command Code								Command Description
C7	C6	C5	C4	C3	C2	C1	C0	
0	0	0	E2	E1	G4	G2	G1	Load data pointer register with contents of E & G fields. (E - 000, G - 110). E & G fields described in Appendix C.
0	0	1	S5	S4	S3	S2	S1	Arm counting for all selected counters
0	1	0	S5	S4	S3	S2	S1	Load contents of specified source into all selected counters
0	1	1	S5	S4	S3	S2	S1	Load & arm all selected counters*
1	0	0	S5	S4	S3	S2	S1	Disarm & save all selected counters
1	0	1	S5	S4	S3	S2	S1	Save all selected counters in hold register
1	1	0	S5	S4	S3	S2	S1	Disarm all selected counters
1	1	1	0	1	N4	N2	N1	Set toggle out (high) for counter N (001 ² N ² 101)
1	1	1	0	0	N4	N2	N1	Clear toggle out (low) for counter N (001 ² N ² 101)
1	1	1	1	0	N4	N2	N1	Step counter N (001 ² N ² 101)
1	1	1	0	1	0	0	0	Set MM14 (disable data pointer sequencing)
1	1	1	0	1	1	1	0	Set MM12 (gate off FOUT)
1	1	1	0	1	1	1	1	Set MM13 (enter 16-bit bus mode)
1	1	1	0	0	0	0	0	Clear MM14 (enable data pointer sequencing)
1	1	1	0	0	1	1	0	Clear MM12 (gate on FOUT)
1	1	1	0	0	1	1	1	Clear MM13 (enter 8-bit bus mode)
1	1	1	1	1	0	0	0	Enable prefetch for write operations
1	1	1	1	1	0	0	1	Disable prefetch for write operations
1	1	1	1	1	1	1	1	Master reset

* Not to be used for asynchronous operations.

Put the hex number 2710 (decimal 10,000) in counter 1 load register:

```
OUT CR, &H9                    'POINT TO COUNTER 1 LOAD REGISTER (TABLE 5-1)
OUT DR, &H10                   'COUNTER 1 LSB
OUT DR, &H27                   'COUNTER 1 MSB
```

Next, set up the counter 2 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = TCN-1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count down
Output Control = TC toggled

VALUE = HEX 0022

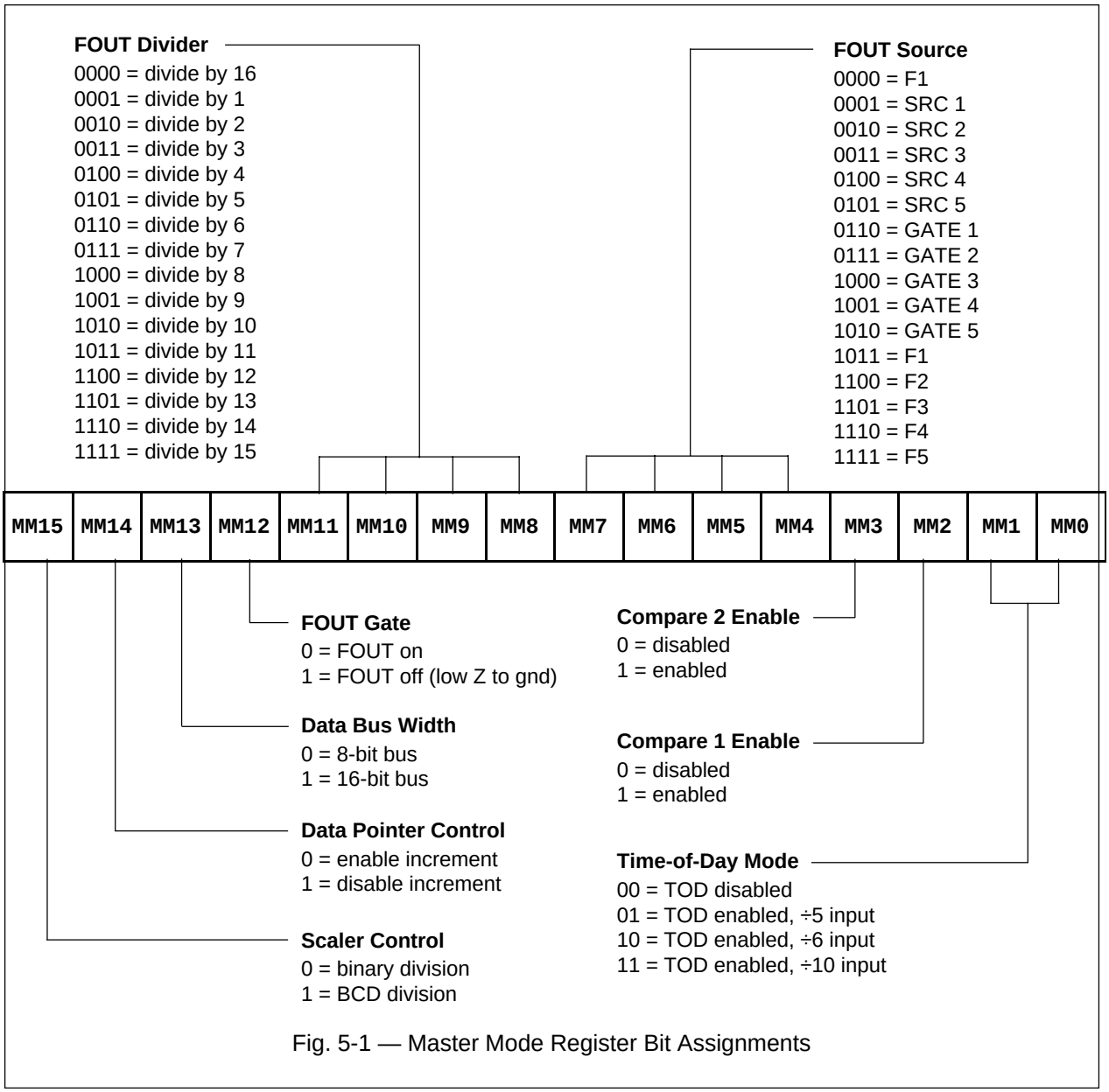

FOUT Divider
0000 = divide by 16
0001 = divide by 1
0010 = divide by 2
0011 = divide by 3
0100 = divide by 4
0101 = divide by 5
0110 = divide by 6
0111 = divide by 7
1000 = divide by 8
1001 = divide by 9
1010 = divide by 10
1011 = divide by 11
1100 = divide by 12
1101 = divide by 13
1110 = divide by 14
1111 = divide by 15
FOUT Source
0000 = F1
0001 = SRC 1
0010 = SRC 2
0011 = SRC 3
0100 = SRC 4
0101 = SRC 5
0110 = GATE 1
0111 = GATE 2
1000 = GATE 3
1001 = GATE 4
1010 = GATE 5
1011 = F1
1100 = F2
1101 = F3
1110 = F4
1111 = F5
MM15 MM14 MM13 MM12 MM11 MM10 MM9 MM8 MM7 MM6 MM5 MM4 MM3 MM2 MM1 MM0
FOUT Gate
0 = FOUT on
1 = FOUT off (low Z to gnd)
Data Bus Width
0 = 8-bit bus
1 = 16-bit bus
Data Pointer Control
0 = enable increment
1 = disable increment
Scaler Control
0 = binary division
1 = BCD division
Compare 2 Enable
0 = disabled
1 = enabled
Compare 1 Enable
0 = disabled
1 = enabled
Time-of-Day Mode
00 = TOD disabled
01 = TOD enabled, ÷5 input
10 = TOD enabled, ÷6 input
11 = TOD enabled, ÷10 input

Fig. 5-1 — Master Mode Register Bit Assignments

```
OUT CR, &H2                 'POINT TO COUNTER 2 MODE REGISTER (TABLE 5-1)
OUT DR, &H22                'COUNTER 2 MODE LSB
OUT DR, &H0                 'COUNTER 2 MODE MSB
```

Put the hex number 1F4 (decimal 500) in counter 2 load register:

```
OUT CR, &HA                 'POINT TO COUNTER 2 LOAD REGISTER (TABLE 5-1)
OUT DR, &HF4                'COUNTER 2 LSB
OUT DR, &H1                 'COUNTER 2 MSB
```

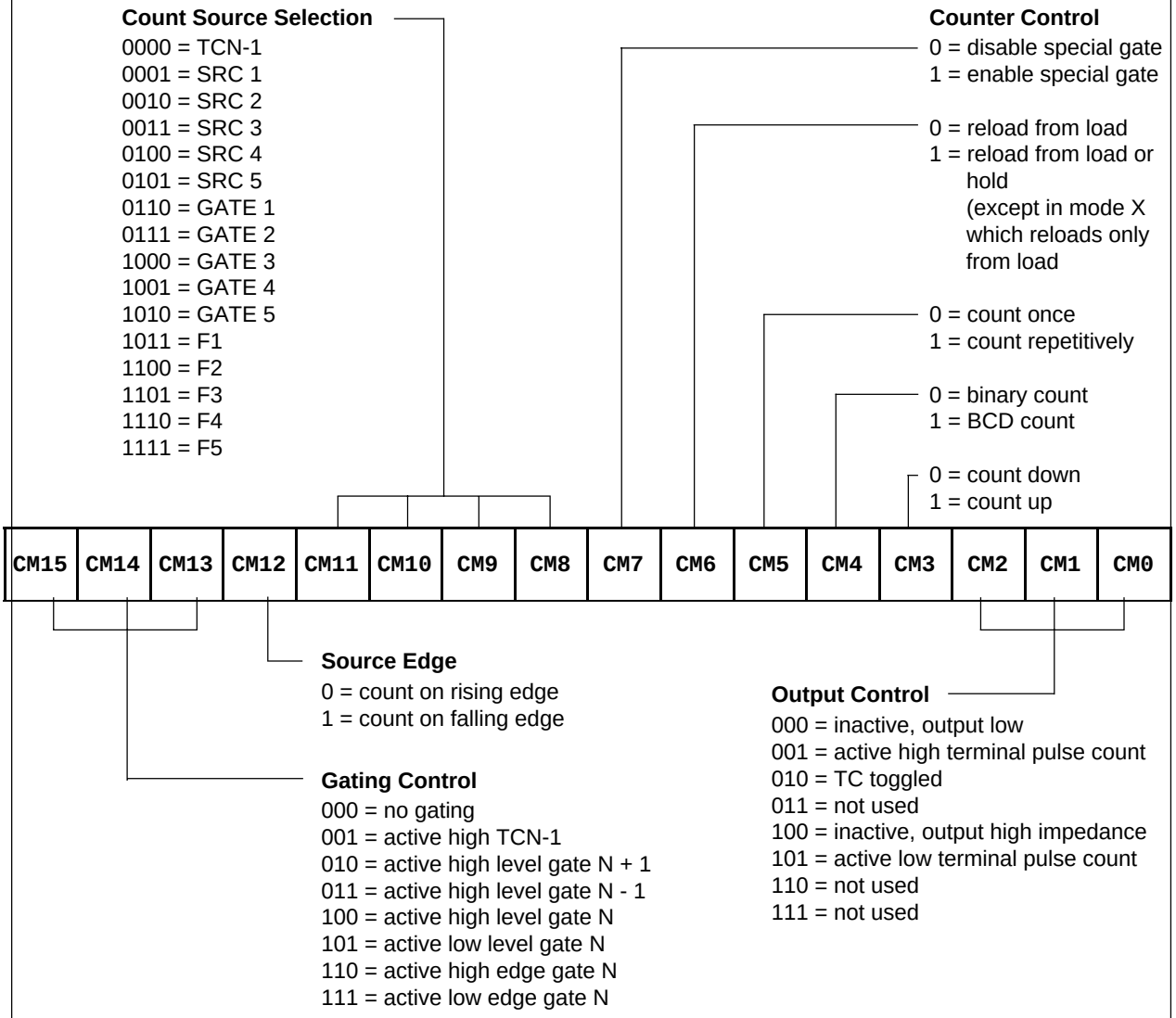

Count Source Selection
0000 = TCN-1
0001 = SRC 1
0010 = SRC 2
0011 = SRC 3
0100 = SRC 4
0101 = SRC 5
0110 = GATE 1
0111 = GATE 2
1000 = GATE 3
1001 = GATE 4
1010 = GATE 5
1011 = F1
1100 = F2
1101 = F3
1110 = F4
1111 = F5
Counter Control
0 = disable special gate
1 = enable special gate
0 = reload from load
1 = reload from load or hold (except in mode X which reloads only from load
0 = count once
1 = count repetitively
0 = binary count
1 = BCD count
0 = count down
1 = count up
CM15 CM14 CM13 CM12 CM11 CM10 CM9 CM8 CM7 CM6 CM5 CM4 CM3 CM2 CM1 CM0
Source Edge
0 = count on rising edge
1 = count on falling edge
Gating Control
000 = no gating
001 = active high TCN-1
010 = active high level gate N + 1
011 = active high level gate N - 1
100 = active high level gate N
101 = active low level gate N
110 = active high edge gate N
111 = active low edge gate N
Output Control
000 = inactive, output low
001 = active high terminal pulse count
010 = TC toggled
011 = not used
100 = inactive, output high impedance
101 = active low terminal pulse count
110 = not used
111 = not used

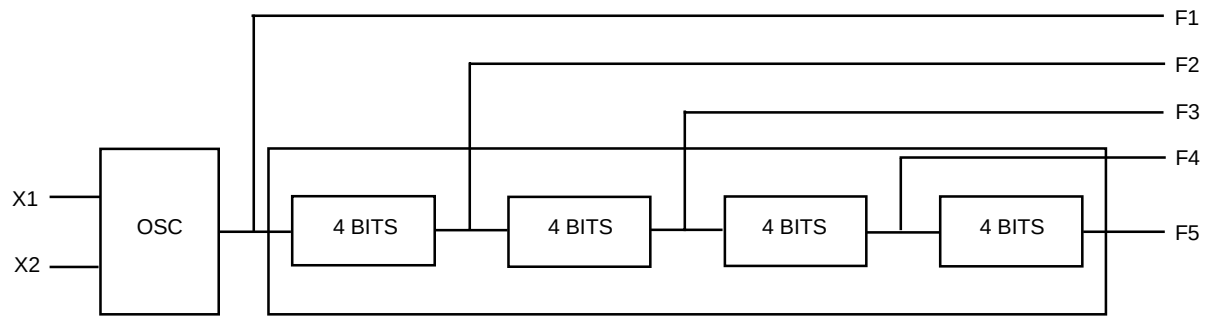
Fig. 5-2 — Counter Mode Register Bit Assignments

Next, set up the counter 3 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = TCN-1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count up
Output Control = TC toggled

VALUE = HEX 002A

```
OUT CR, &H3                    'POINT TO COUNTER 3 MODE REGISTER (TABLE 5-1)
OUT DR, &H2A                   'COUNTER 3 MODE LSB
OUT DR, &H0                    'COUNTER 3 MODE MSB
```



Frequency	BCD Scaling (MM15 = 1)		Binary Scaling (MM15 = 0)	
	Ratio	With On-board 5 MHz Clock	Ratio	With On-board 5 MHz Clock
F1	OSC	5 MHz	OSC	5 MHz
F2	F1 / 10	500 kHz	F1 / 16	312.5 kHz
F3	F1 / 100	50 kHz	F1 / 256	19.53 kHz
F4	F1 / 1000	5 kHz	F1 / 4096	1.221 kHz
F5	F1 / 10,000	500 Hz	F1 / 65,536	76.3 Hz

Fig. 5-3 — Frequency Scaler Ratio

Put the hex number 0000 in counter 3 load register:

```
OUT CR, &HB                    'POINT TO COUNTER 3 LOAD REGISTER (TABLE 5-1)
OUT DR, &H0                    'COUNTER 3 LSB
OUT DR, &H0                    'COUNTER 3 MSB

OUT CR, &H67                   'LOAD & ARM COUNTERS 1,2 & 3 (TABLE 5-2)
```

The main program is:

```
OUT CR, &HA4                   'SAVE COUNTER 3 IN HOLD REGISTER (TABLE 5-2)
OUT CR, &H13                   'POINT TO COUNTER 3 HOLD REGISTER (TABLE 5-1)
LSB = INP(DR)                  'READ COUNTER 3 LSB
MSB = INP(DR)                  'READ COUNTER 3 MSB
RESULT = LSB + (MSB * 256)     'COMBINE LSB & MSB
LOCATE 12, 37
PRINT USING "#####"; RESULT
ON KEY(1) GOSUB QUIT
GOTO MAIN
```

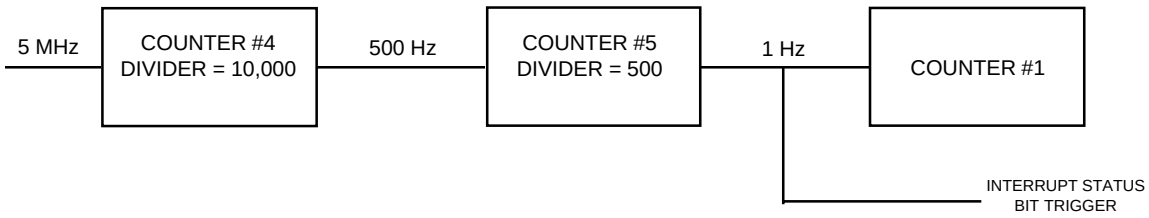
To quit:

```
KEY(1) OFF
END
```

EXAMPLE: Setting Up the Am9513A as a Frequency Counter

This program, FCOUNT on the example disk, shows you how to program the Am9513A as a simple frequency counter. In this example, counter 4 is used to divide the on-board 5 MHz clock by 10,000. The output from counter 4 (5 MHz ÷ 10,000 = 500 Hz) is used to clock counter 5. Counter 5 is used to divide this 500 Hz clock by 500. The result is a 1 Hz output which is used to gate counter 1. This 1 Hz output is also used to trigger the interrupt status bit. The program monitors this status bit to determine when to read counters 1 and 2. Counters 1 and 2 are used to count the SRC1 input for 1 second intervals.

NOTE: For this program to operate properly, a jumper must be connected between OUT5 (pin 16) and GATE1 (pin 2) at I/O connector P2.



The first lines of the program clear the screen and set up the base address of the DM5804. The address in the variable "BA" must match the setting of the base address switch, S1, on the board. The factory setting of S1 is 300 hex (768 decimal).

```
CLS
INPUT "ENTER BASE ADDRESS IN DECIMAL: "; BA
```

The next section of the program sets up the computer screen:

```
CLS                              'CLEAR SCREEN
DIM RESULT AS DOUBLE             'DIMENSION VARIABLE "RESULT"
KEY(1) ON                        'TURN F1 KEY ON
LOCATE 2, 25
PRINT "DM5804 FREQUENCY COUNTER DEMO PROGRAM !";
LOCATE 24, 2
PRINT "F1 = QUIT ";
```

The next section of the program sets up the address for the DM5804 registers. These addresses are defined at the beginning of Chapter 4.

```
PA = BA + 0                      'ADDRESS FOR 8255 PORT A
PB = BA + 1                      'ADDRESS FOR 8255 PORT B
PC = BA + 2                      'ADDRESS FOR 8255 PORT C
CW = BA + 3                      'ADDRESS FOR 8255 CONTROL WORD
DR = BA + 4                      'ADDRESS FOR AM9513A COUNTER DATA REGISTER
CR = BA + 5                      'ADDRESS FOR AM9513A COUNTER CONTROL REGISTER
IRQEN = BA + 6                   'ADDRESS FOR INTERRUPT ENABLE
STAT = BA + 7                    'ADDRESS TO READ INTERRUPT STATUS
IRQCLR = BA + 7                  'ADDRESS TO CLEAR INTERRUPT STATUS BIT
```

Next, you must enable status generation:

```
OUT IRQEN, 1
```

Now, reset the Am9513A timer/counter chip (see Table 5-2):

```
OUT CR, &HFF                     'AM9513A MASTER RESET
```

Next, set up the Am9513A master mode register (see Figure 5-1). These are the settings we will use:

Scaler Control = binary division
Data Pointer Control = enable increment
Data Bus Width = 8 bits
FOUT Gate = FOUT on
FOUT Divider = divide by 16
FOUT Source = F1 (see Figure 5-3)
Compare 2 Enable = disabled
Compare 1 Enable = disabled
Time-of-Day Mode = disabled

VALUE = HEX 0000

```
OUT CR, &H17                    'POINT TO MASTER MODE REGISTER (TABLE 5-1)
OUT DR, &H0                     'MASTER MODE LSB
OUT DR, &H0                     'MASTER MODE MSB
```

Next, set up the counter 1 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = active low gate n
Source Edge = rising edge
Count Source Selection = SRC1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count up
Output Control = active high TC

VALUE = HEX A129

```
OUT CR, &H1                     'POINT TO COUNTER 1 MODE REGISTER (TABLE 5-1)
OUT DR, &H29                    'COUNTER 1 MODE LSB
OUT DR, &HA1                    'COUNTER 1 MODE MSB
```

Put the hex number 0000 (decimal 0) in counter 1 load register:

```
OUT CR, &H9                     'POINT TO COUNTER 1 LOAD REGISTER (TABLE 5-1)
OUT DR, &H0                     'COUNTER 1 LSB
OUT DR, &H0                     'COUNTER 1 MSB
```

Next, set up the counter 2 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = TCN-1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count up
Output Control = active high TC

VALUE = HEX 0029

```
OUT CR, &H2                     'POINT TO COUNTER 2 MODE REGISTER (TABLE 5-1)
OUT DR, &H29                    'COUNTER 2 MODE LSB
OUT DR, &H0                     'COUNTER 2 MODE MSB
```

Put the hex number 0000 (decimal 0) in counter 2 load register:

```
OUT CR, &HA                    'POINT TO COUNTER 2 LOAD REGISTER (TABLE 5-1)
OUT DR, &H0                    'COUNTER 2 LSB
OUT DR, &H0                    'COUNTER 2 MSB
```

Next, set up the counter 4 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = F1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count down
Output Control = TC toggled

VALUE = HEX 0B22

```
OUT CR, &H4                    'POINT TO COUNTER 4 MODE REGISTER (TABLE 5-1)
OUT DR, &H22                   'COUNTER 4 MODE LSB
OUT DR, &H0B                   'COUNTER 4 MODE MSB
```

Put the hex number 2710 (decimal 10,000) in counter 4 load register:

```
OUT CR, &HC                    'POINT TO COUNTER 4 LOAD REGISTER (TABLE 5-1)
OUT DR, &H10                   'COUNTER 4 LSB
OUT DR, &H27                   'COUNTER 4 MSB
```

Next, set up the counter 5 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = TCN-1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count down
Output Control = TC toggled

VALUE = HEX 0022

```
OUT CR, &H5                    'POINT TO COUNTER 5 MODE REGISTER (TABLE 5-1)
OUT DR, &H22                   'COUNTER 5 MODE LSB
OUT DR, &H0                    'COUNTER 5 MODE MSB
```

Put the hex number 1F4 (decimal 500) in counter 5 load register:

```
OUT CR, &HD                    'POINT TO COUNTER 5 LOAD REGISTER (TABLE 5-1)
OUT DR, &HF4                   'COUNTER 5 LSB
OUT DR, &H1                    'COUNTER 5 MSB

OUT IRQCLR, 0                  'CLEAR INTERRUPT STATUS
OUT CR, &H7B                   'LOAD & ARM COUNTERS 1,2,4 & 5 (TABLE 5-2)
```

The main program is:

```
IRQ = INP(STAT) AND 1
IF IRQ <> 1 GOTO MAIN           'CHECK IF INTERRUPT STATUS = 1
OUT IRQCLR, 0                   'CLEAR INTERRUPT STATUS
OUT CR, &H83                    'DISARM & SAVE COUNTERS 1 & 2 (TABLE 5-2)
OUT CR, &H63                    'LOAD & ARM COUNTERS 1 & 2 (TABLE 5-2)
OUT CR, &H11                    'POINT TO COUNTER 1 HOLD REGISTER (TABLE 5-1)
LSB(1) = INP(DR)                'READ COUNTER 1 LSB
MSB(1) = INP(DR)                'READ COUNTER 1 MSB
LSB(2) = INP(DR)                'READ COUNTER 2 LSB
MSB(2) = INP(DR)                'READ COUNTER 2 MSB
RESULT = LSB(1) + (MSB(1) * 256) + (LSB(2) * 256 ^ 2) + (MSB(2) * 256 ^ 3)
LOCATE 12, 35
PRINT USING "##########"; RESULT;
PRINT "Hz"
ON KEY(1) GOSUB QUIT
GOTO MAIN
```

To quit:

```
KEY(1) OFF
END
```

APPENDIX A

DM5804/DM6804 SPECIFICATIONS

DM5804/DM6804 Characteristics Typical @ 25° C

Interface

cpuModule™ & other PC/104 form factor compatible
Switch-selectable base address, I/O mapped
Jumper-selectable interrupts

Digital I/O ..... CMOS 82C55 (Optional NMOS 8255)

Number of lines 24
Logic compatibility ..... TTL/CMOS
(Configurable with optional I/O pull-up/pull-down resistors)
High-level output voltage ..... 4.2V, min
Low-level output voltage ..... 0.45V, max
High-level input voltage ..... 2.2V, min; 5.5V, max
Low-level input voltage ..... -0.3V, min; 0.8V, max
Input load current ..... ±10 µA
Input capacitance,
C(IN)@F=1MHz ..... 10 pF
Output capacitance,
C(OUT)<@F=1MHz ..... 20 pF

Timer/Counter Am9513A

Five 16-bit timer/counters
Binary or BCD up or down counting
Programmable operating modes 24
Counter input source ..... External clock (6.9 MHz, max); on-board 5 MHz clock; external gate input; or adjacent counter output
Counter outputs ..... Available externally; used as PC interrupts or internally cascaded to adjacent counter
Counter gate source ..... External input; counter output; or software control

Miscellaneous Inputs/Outputs

+5 volts/digital ground (PC bus-sourced)
External interrupt input
Frequency output
Additional gate inputs

Current Requirements

+5 volts ..... 220 mA, max

P2 Connector

50-pin right angle header

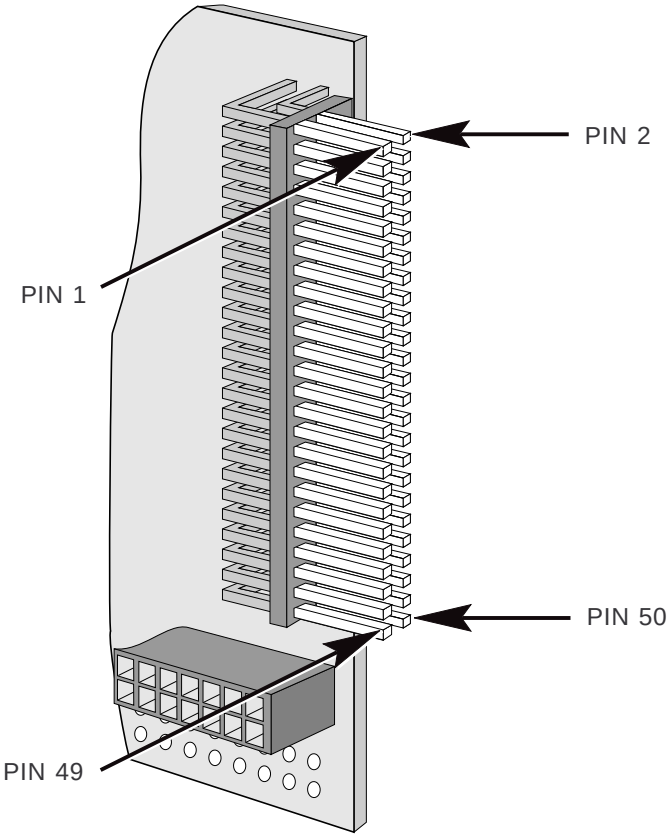
Size

3.55"L x 3.775"W x 0.6" H (90mm x 96mm x 16mm)

APPENDIX B

P2 CONNECTOR PIN ASSIGNMENTS

SRC1	1	2	GATE1
OUT2	3	4	OUT1
GATE2	5	6	SRC2
SRC3	7	8	GATE3
OUT4	9	10	OUT3
GATE4	11	12	SRC4
SRC5	13	14	GATE5
DIGITAL GND	15	16	OUT5
EXTINT	17	18	DIGITAL GND
FOUT	19	20	DIGITAL GND
DIGITAL GND	21	22	DIGITAL GND
PA7	23	24	PC7
PA6	25	26	PC6
PA5	27	28	PC5
PA4	29	30	PC4
PA3	31	32	PC3
PA2	33	34	PC2
PA1	35	36	PC1
PA0	37	38	PC0
PB7	39	40	PB6
PB5	41	42	PB4
PB3	43	44	PB2
PB1	45	46	PB0
+12 VOLTS	47	48	+5 VOLTS
-12 VOLTS	49	50	DIGITAL GND



P2 Mating Connector Part Numbers	
Manufacturer	**Part Number**
AMP	1-746094-0
3M	3425-7650

APPENDIX C

COMPONENT DATA SHEETS

AMD Am9513A System Timing Controller
Data Sheet Reprint

Intel 82C55A Programmable Peripheral Interface
Data Sheet Reprint

APPENDIX D

WARRANTY

LIMITED WARRANTY

Real Time Devices, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from REAL TIME DEVICES. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, REAL TIME DEVICES will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to REAL TIME DEVICES. All replaced parts and products become the property of REAL TIME DEVICES. **Before returning any product for repair, customers are required to contact the factory for an RMA number.**

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by REAL TIME DEVICES, “acts of God” or other contingencies beyond the control of REAL TIME DEVICES), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN REAL TIME DEVICES. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND REAL TIME DEVICES EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER’S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL REAL TIME DEVICES BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

DM5804 Board User-Selected Settings	
Base I/O Address:	
(hex)	(decimal)
IRQ Channel Selected (Select ONE Interrupt Source, P4):	
PC3	IRQ Channel #:
PC0	IRQ Channel #:
OUT5	IRQ Channel #:
EXTINT	IRQ Channel #: